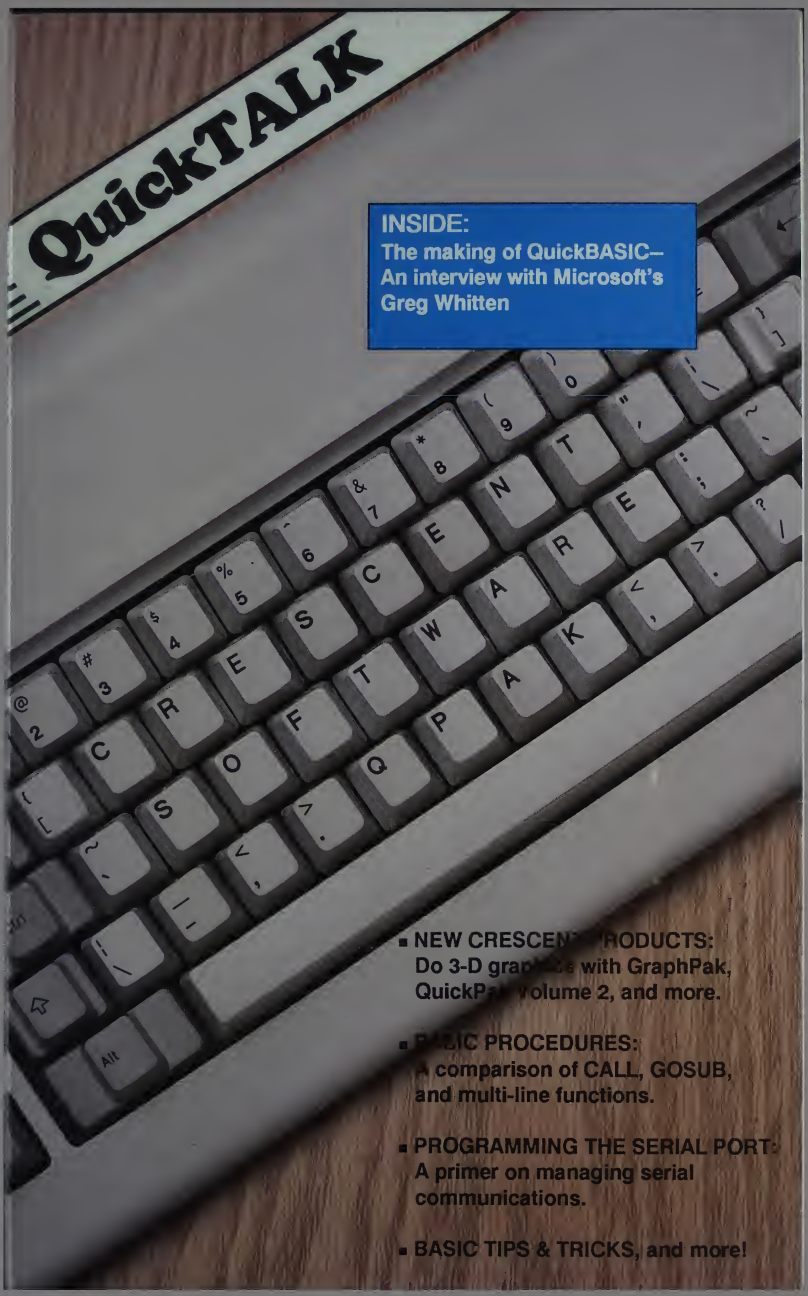

QuickTALK
INSIDE:
The making of QuickBASIC–
An interview with Microsoft's
Greg Whitten
■ NEW CRESCENT PRODUCTS:
Do 3-D graphics with GraphPak,
QuickPak Volume 2, and more.
■ BASIC PROCEDURES:
A comparison of CALL, GOSUB,
and multi-line functions.
■ PROGRAMMING THE SERIAL PORT:
A primer on managing serial
communications.
■ BASIC TIPS & TRICKS, and more!

Table of Contents

QuickTALK is published whenever we get around to it by Crescent Software.
Volume 1, Number 1

Written by Ethan Winer
Contributing writers : Nash Bly, Jay Munro, Sureta Escobar
Design, photography, cartooning and whatever: Jay Munro
Support & Spiritual Guidance: Sureta Escobar

Crescent Sortware
64 Fort Point St
Norwalk, CT 06855
203-846-2500

A message from the publisher

Welcome to the premier issue of QuickTALK — Crescent Software's magazine for BASIC programmers. More than simply a catalog of our products, QuickTALK includes feature articles, tips and tutorials, and provides newcomers with an entertaining way to get to know us. This magazine also commemorates our first year in business.

...we want you to understand how our programs work.

In twelve short months we have grown tremendously, beginning with a single offering—QuickPak—to a collection of five products that support BASIC. Beyond simply offering useful programs and utilities, we have also come to be known as a source for assistance and sound technical advice.

We realize that customer satisfaction depends not only on a quality product, but the support effort behind that product. Where some companies charge extra for source code, we *want* you to understand how our programs work. Freely sharing what we know and caring about our customers has been the foundation of our success.

Of course, great software at reasonable prices is something our customers have come to expect. QBase would be a terrific value as a screen builder alone, but we also include a complete relational database. At $69, QuickPak is a bargain just for the routines and example programs, but we throw in commented source code, an assembly language tutorial, and a BASIC tips and tricks book as well. Even this magazine reflects our philosophy—it's free!

Why BASIC?

BASIC is not only the easiest programming language to learn and use, but is the hands-down winner in sheer capabilities. As Microsoft and Borland battle it out to create ever faster and better BASIC compilers, the *real* winner is unquestionably you.

BASIC for personal computers has grown from a slow and limited language interpreter to a powerful programming and development environment. With the advent of QuickBASIC version 4, we have the ability to call routines written in other Microsoft languages, create record structures and user-definable variable types, and much more. BASIC has indeed come of age.

But BASIC is also getting more complicated. True, you don't need to use all of these new features to program effectively, however they offer many truly exciting capabilities. We are committed to staying abreast of current and future BASIC releases, and will continue to share our insight and experience with our customers.

—Ethan Winer

Who's.....Who

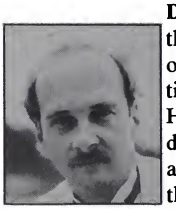
Our Founder

ETHAN WINER founded Crescent Software in 1986 after serving for many years as a consultant to several major corporations. He is a self-taught expert in BASIC, DOS, and assembly language. When not writing programs or conducting business, Ethan writes feature articles for PC Magazine, Borland's Turbo Technix, Programmer's Journal, and IBM's Exchange Magazine.

Prior to his fascination with computers, he performed professionally as a musician (guitar and electric bass). Ethan is also credited with designing and building one of Connecticut's major recording studios. He has composed, arranged, and performed on many radio and TV commercials, and is an avid electronics enthusiast. Ethan has also built several music synthesizers and other related gadgets.

SURETA ESCOBAR joined the Crescent team in February of 1987 as manager of operations. Regardless of what Ethan might like to think, it is Sureta who *really* runs the show. An indispensable part of the business, she maintains the daily work flow, and edits and co-writes the Crescent advertising and promotional literature.

When not assisting customers or playing with a computer, her talents take a different turn. Sureta teaches classes in both Astrology and Parapsychology, and does readings professionally.



DON MALIN is the brains behind our QBase relational database. He came to us one day with an idea and a demo, and the rest is history. Besides being just about the best BASIC programmer we know, Don is also a Lotus 123 expert. He has written and markets a collection of templates and BASIC programs for owners of offshore racing yachts. Don was part of the team that built the racing boat Freedom, which won the America's Cup competition.

WHO'S WHO Cont.

JAY MUNRO is Crescent's resident photographer and cartoonist. (Doesn't *every* company need a resident cartoonist?) He's also our favorite person. Jay has a degree in cinematography, and worked for four years as photo-A/V coordinator for Save The Children Federation. When he's not helping us out, you can find him providing customer support for dOTHER big database company.

BRIAN GIEDT is the creator of GraphPak, and he's one of the most amazing guys around–you'll need exponential notation to count the number of computer languages he knows. Brian has extensive AI experience, he's taught college level courses in programming, and is currently developing a powerful hypertext word processor.

In anticipation of sales figures exceeding $200 billion in 1987, Crescent Software prepares to move into new corportate headquarters.

PROGRAMMING FOR SERIAL COMMUNICATIONS

by Nash Bly

If you've ever needed to tame the COM ports from a BASIC program, these tips from a communicatons expert will really help.

BASIC programmers who intend to use the serial communication ports to access external devices must first have an understanding of the data format and data rate that will be used. This is true from a data handling point of view (the programmer's role) and from a serial interface point of view (the RS-232 specialist's role).

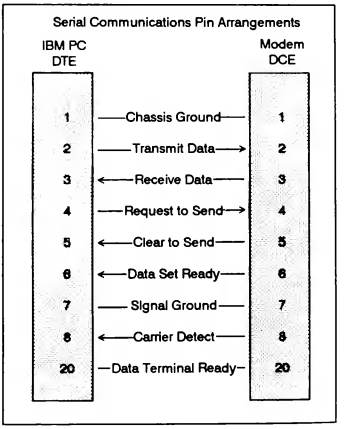
The first player must consider commas, quote marks, carriage returns, line feeds, line lengths, field types, the amount of data to process, and the time available for processing. The second player must consider stop bits, data bits, baud rate, error checking, handshaking, and the type of cable being used. If these two roles are the part you're playing, then read on.

GETTING STARTED

Once you've written a program and connected the equipment, if it doesn't work immediately, it's difficult to know where the problem lies. Is it the hardware, the software, or perhaps the wrong type of cable? Therefore, before you begin writing any code at all involving the COM port, it's a good idea to first use an existing communications program.

A program such as PC-Talk will work fine, with or without a modem. If you are using a modem, then you're probably already connected and tested, so skip ahead. But if you're connecting to some other type of device, then the appropriate technical information on that device will help you choose the proper connecting cable.

There are two different wiring configurations for the standard DB-25 connector, which are designed to connect together–DTE (Data Terminal Equipment) and DCE (Data Communications Equipment). These are shown below.



A computer such as the PC has a DTE pin arrangement, while a modem uses DCE. The connection between a DTE and a DCE uses a standard cable which is wired pin for pin (though not all 25 pins are actually required). That is, pin 2 at one end connects to pin 2 at the other, and so forth.

Connections between devices using like pin arrangements (eg. DTE to DTE) require a modem eliminator. This is a cable with the appropriate pins reversed, or an adaptor added to a standard cable. For example, pin 2 (send) at one end is connected to pin 3 (receive) at the other. The key point is that these devices will be both sending and receiving data, and the correct wiring arrangement must be observed.

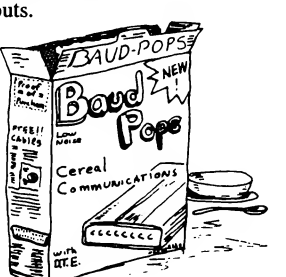
Once the cable is ready, you must match the type of parity checking, baud rate, number of data bits, and number of stop bits used by the device and the communication program. Normally, matching is done by setting communication parameters used by the program to values identical to those used by the device. For example, in PC-Talk these values are accessed with the ALT-F or Alt-P commands. The most common baud rates used for controlling equipment are 1200, 9600, and 300 respectively.

HANDSHAKING

Control of data flow can be accomplished with either hardware or software. The hardware method is referred to as *handshaking*. Handshaking requires the DTE to have the go-ahead from the DCE before sending data. This is done by an exchange of control signals over the cable. In cases where a device does not generate or receive the expected signals, it will be necessary to "hot-wire" the DTE to get things rolling.

If you believe the DTE device is not sending, disconnect the wires from pins 4 and 5 at its end, and then connect a jumper between the connector pins. This will guarantee a go-ahead, even if the receiving device does not know enough to request data.

Likewise, the serial port in a PC is sensitive to the signal on pin 6. If this pin is not high, the PC will think the device is not ready for action. In BASIC, software can over-ride hardware handshaking, though some communication programs may not be as flexible. In these cases you must jumper the cable pins as described above, to satisfy the data control inputs.



PROTOCOL

The software method for controlling data flow is called XON/XOFF Protocol. This too can block the succesful transfer of data if it is not correct. XON/XOFF requires the receiving device—whether DTE or DCE—to send a start character when it is ready to receive, and a stop character when it needs to pause. When all is working correctly, the sending device will not send data until a start character is received, and it will stop sending when it receives a stop character.

The start character is Ctrl-Q or Chr$(17), and the stop character is Ctrl-S or Chr$(19). Communications programs such as PC-Talk will disable this protocol if you tell them to do so. Of course, you should always disable XON/XOFF to transfer data to or from devices that don't use this protocol, and enable it for devices that do.

TEST DATA

If you are writing a program to send from the COM port, you should experiment by preparing some test data with a word processor, and then send it via your communications program. Programming to receive data from the COM port requires knowing what ASCII characters will be sent, and whether they will come in a large bunch or in small groups. Prepare the device to send some test data to your communications program so you may examine it. Test data received by PC-Talk is conveniently stored character for character onto disk.

When examined by a word processor, certain characters may not appear, even though they exist. This is because null characters, control characters, and leading spaces are stripped out by many word processors. Some word processors are also incapable of sending control characters. To solve these problems, I often use Sidekick's notepad, which has the feature that all characters will be shown, and *any* character can be sent. This is especially important with input routines, since space and null characters received but not accounted for will cause problems.

Continued next page

TIP:

Want to know the fastest way to clear a string to all blanks without actually erasing it? Easy:

```
LSET Test$ = ""
```

BASIC's COM1:

As you may have already discovered, BASIC handles communications very much like a disk file. It must be opened and closed, and it is identified with a file number during input and output. A communications buffer to hold the incoming and outgoing data is created in memory when the COM port is opened. The transmission buffer is 128 bytes, and by default the receiving buffer is 256 bytes. Once the port has been opened, data exchange is accomplished using the familiar disk I/O statements.

The receiving buffer can be increased with the /C option, which means that additional memory is set aside to hold the incoming data. This may be necessary if your program will be performing other operations while data is being received. If the program can't (or doesn't) read in the data as fast as it comes in, the buffer will overflow and the excess will be lost. Using On Error and Resume may prevent visible problems, but the data will be gone none the less.

OPENING THE PORT

The Open Com statement allows you to control the communication parameters, handshaking, protocol, and a few other details. Unidirectional/sequential or bidirectional/random access can be selected using the Mode option. Note that you must pay special attention to the use of commas for correct syntax when using Open Com.

It may be helpful to start out with no handshaking, and then, once you have completed debugging your program, you may elect to reestablish handshaking. Once this is done, however, you must be prepared to handle the errors that will result when the device handshaking with the PC is not ready.

To disable handshaking, include the RS option and set the timeout delay to 0 for the CS, DS, and CD options. Setting the CS option to 0 will act as a jumper between pins 4 and 5, as described earlier. Setting the DS option to 0 means that pin 6 will be ignored. The CD (carrier detect) option should be non-zero only when using a modem. A minimal discussion of these parameters is given in the QuickBASIC documentation.

When deciding whether to open the port in the binary (hardware handshaking) or ASCII (XON/XOFF) mode, consider that the binary mode can do anything that the ASCII mode can, with a little help from you. For example, if you are interested in using XON/XOFF but don't want Ctrl-Z to be an end of file marker, simply write your own code for sending and responding to Chr$(17) and Chr$(19).

SENDING DATA

If the port is operating in the sequential mode, sending data is simply a matter of executing a Print# or Write# statement. Print# will most likely be your choice not only because it's simpler, but also because double quotations can be part of a string to be sent with Print#.

In random access mode, don't include a record number in the Put# statements because there aren't any records associated with the COM port like there are in a disk file. Do, however, convert all data to be sent to strings, and place them in a random access file buffer with Field, RSet, and LSet statements.

If the receiving device wants to control the PC's output with XON/XOFF, then random access mode may be required so that Ctrl-S and Ctrl-Q can be detected while sending data out. If XON needs to be detected only once at the start of transmission, use a sequential Open for input to detect XON, followed by a Close, and then re-open the port again for Output.

Sometimes a receiving device will time-out and stop receiving if too much time passes between one output statement and the next. Therefore, once data transmission starts, periods of inactivity at the COM port must be kept short. You must process a little, then send a little. If enough memory is available, an array may be filled with the data before transmission begins. A simple loop can then transmit all the data in one continuous action.

RECEIVING DATA

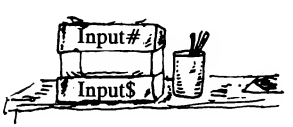
Keeping track of how many characters are in the communications buffer and anticipating what those characters are is the key to a successful input routine. Attempting to input from an empty buffer results in an Input Past End error, while ignoring a buffer filling with data will cause a Buffer Overflow error. Two methods are available in BASIC for managing these problems.

The first method uses the LOF(#), EOF(#) and LOC(#) functions to monitor the status of the buffer. A program checks to see if data is in the buffer, and if so moves it into memory. Often, once data is in the buffer, a program will continue executing input statements and checking the buffer until it is empty, suspending other operations. If the computer is unable to check or act upon the status of the buffer during other operations, the buffer must be large enough to hold the incoming data until the completion of that operation.

The second method, event trapping, uses the Com On and On Com statements. As soon as any data enters the buffer, your program will automatically branch to a subroutine which handles the data reception. This may be very convenient in some cases, but it can easily become problematic.

Branching to the subroutine will occur even if just a few insignificant characters are in the buffer. This can lead to a freeze-up, because an input statement will simply wait until a terminating character is received. Other input activities can be difficult if not impossible while using On Com, plus library routines or subprograms outside the main compiled module cannot access the event handling subroutines. It may be better to monitor the buffer yourself, as described above.

Wait until there are enough characters to insure that data is present before executing an Input statement. Your choice of input statements will depend on the data format that is being received. If carriage returns are included, Line Input# should be used since it doesn't stop on commas or quotes, and insignificant leading characters are automatically stripped.



Otherwise, Input# and Input$ must be used. This is where examination of test data comes in handy. XON/XOFF can be implemented by simply sending the appropriate characters. If your program can keep up with the incoming data, XOFF may never need to be sent. First open the port for output, send XON, close the port, and finally re-open it for input. In this way, maintenance of a random access file buffer can be avoided.

Nash Bly is a freelance programmer and video communications engineer.

How about that!

BASIC sure has accumulated quite a few new key words lately. There's Select and Case, Loop, Until, and who knows how many more. Enough already! How about forgetting all the new stuff, and improving some of the existing commands. Some changes you'll probably never see include:

Currently	Should Be
Absolute	Absolut
Dynamic	Bodacious
Error	BooBoo
Get	Gimme
If/Then/Else	If/Then/NeverMind
Input	Hit Me

Tip:

As far as we're concerned, the crime of the century is the fact that Microsoft put all kinds of capabilities into LINK and BUILDLIB, but failed to mention them anywhere in the QuickBASIC documentation. My favorite undocumented LINK option is /E, which creates a special packed file that can be as much as 20% smaller than it would be without the option. Compile your program as usual, and then start LINK like this:

```
LINK program /E
```

Not unlike the various ARC utilities, /E packs your program code and data using a special algorithm. Then at run time, the first code that actually executes is the unpacking routine which puts your program back in order, and then runs it. Neat, right?

BUILDLIB has a few hidden goodies too, for example the /L option will bring in all of the modules in a .LIB file. If you want to combine your subprograms with the various assembler routines from one or more Crescent .LIB files, simply start BUILDLIB like this:

```
BUILDLIB prog1 + prog2 + prog3 . . . /L
```

When BUILDLIB prompts for a library, enter the names of all the .LIB files you want to include separated by blanks. One warning, though, is when combining several Crescent .LIB libraries. Since some of the routines appear in more than one library, BUILDLIB may give an error message. You can ignore this, because the final user library will still operate correctly.

Speaking of packed files, don't forget that you can chop about 1K off the total size of a stand-alone program by including the SMALLERR.OBJ file when you link it. Most programmers use their own error messages anyway, and don't need QuickBASIC's too.

```
Play "MNL8O3F<AA>FL2FL8<AA>FF<AB-AGGG>EL4E"
```

Bugs and new additions:

What do death, taxes, and program bugs have in common? They're all inevitable! Here's a few we have identified and corrected. If you have a Crescent product listed below, simply return your original disk in a pre-paid mailer with a note, and we'll send you a new one pronto. Otherwise, many of the problems listed below can be corrected by simply making the appropriate changes as indicated.

- QuickPak version 12 has a problem with the ReadScrn routine (in the \BONUS directory) that causes a crash when used on a CGA display.

- QuickPak versions prior to 14 contain a bug in the assembler string sort routine, where occasionally two of the array elements would be out of order. (Special thanks to Bob Lincoln for reporting this and providing the fix!)

- Early copies of QuickPak for Turbo Basic have a bug in the FileInfo subprogram, where certain file times cause an Illegal Function Call error.
  That portion of the code SHOULD read:

```
X = Peek(DTAddr! + 22) + 256! * Peek(DTAddr! + 23)      'the time
X = Int(X / 2)                                          'prevent overflow
Info(3) = (X And &B1111110000000000) \ 1024             'the hour
Info(4) = (X \ 16) And &B111111                         'the minute
Info(5) = (X And &B1111) * 4                            'the second
```

- The QuickPak routines Bar2Menu and PullDown both contain a line label Display:. Of course, there's no problem unless you're using both routines in one program. Otherwise, simply change one of them.

- The QuickPak manual shows the arguments for PUsing.Fn reversed. The quick reference card is correct, but the manual example should read:

```
Print FnPUsing$(Mask$, Value#)
```

- The Caps.On and Num.Only arguments to the QuickPak 2 Editor routine are shown reversed in the manual.

We've also added a set of routines to QuickPak for Turbo Basic to operate an EGA in the 43 line text mode. If you want this feature, return your original disk and we'll mail you the latest version.

Copying Files in Basic

One of the really nifty things about Turbo Basic is its Binary file capability, which lets you read or write to *any* portion of a disk file. We've added Binary file routines in QuickPak 2 for QuickBASIC, plus an example program that allows you to copy files without needing SHELL or COMMAND.COM. Here's how you can do it in Turbo Basic.

Like the DOS copy command, this subprogram lets you specify a destination drive or path only, or optionally use a new name.

```
'********** DemoCopy - copies files in a Turbo Basic program

'Copyright (c) 1987 Ethan Winer

Defint A-Z
Cls
Line Input "File to copy: ", InFile$              'prompt for the source file name
Line Input " Destination: ", OutFile$             'ditto for the target and/or path

Call FileCopy(InFile$, OutFile$)                  'do the copy
End


 Sub FileCopy(Source$, Dest$) Static

   Local Buffer$                                  'just need this temorarily

   If Right$(Dest$, 1) = ":" Or _                 'see if only a drive or
     Right$(Dest$, 1) = "\" Then                  ' path was given

     For X = Len(Source$) To 1 Step -1            'isolate the source name
       If Mid$(Source$, X, 1) = ":" Or _          ' from the drive or path
         Mid$(Source$, X, 1) = "\" Then           ' if appropriate
         Dest$ = Dest$ + Mid$(Source$, X + 1)     'keep only name portion
         Exit For                                 'terminate For/Next loop
       End If                                     ' and set X as a flag,
     Next                                         ' X will be 0 otherwise
     If X = 0 Then Dest$ = Dest$ + Source$        'the source had no drive
   End If                                         ' or path append it to

   Open Source$ For Binary As #99                 'open the source file, 99 is unlikely
      Size! = Lof(99)                             ' to conflict with your programs

   Open Dest$ For Binary As #100                  'ditto for the destination file

   FreeMem! = Fre(" ") - 1000                     'see how big a buffer we can use
```

Democopy Continued

```
  If FreeMem! > 32767 Then _          'oops, we can't make it THAT big!
    FreeMem! = 32767

  If Size! = > FreeMem! Then          'now see how much we have to copy
    BufSize = FreeMem!                'copy the file in increments
  Else
    BufSize = Size!                   'small enough to copy in one shot
  End If

  Location! = 0                       'start reading at the beginning

More:

  Seek #99, Location!                 'seek to the same location in each file
  Seek #100, Location!

  Get$ #99, BufSize, Buffer$          'fill Buffer$ with the source data
  Put$ #100, Buffer$                  'put it into the destination file
  Buffer$ = ""                        'free up the memory for a moment

  Location! = Location! + BufSize     'increment pointer to the next portion of the
                                      '    source
  Size! = Size! - BufSize             'track how much remains

  If Size! = > BufSize Goto More      'at least BufSize bytes left, go do it
  If Size! > 0 Then                   'still more, but less than BufSize
    BufSize = Size!
    Goto More
  End If

  Close #99, #100                     'all done, close the files

End Sub
```

An interview with Microsoft's Greg Whitten

If you've ever wondered where the "GW" in GWBASIC comes from, wonder no more—some of the real genius behind the development of Microsoft BASIC, BASCOM, and QuickBASIC is Greg Whitten.

Greg was kind enough to take time away from his busy schedule as Director of Languages R&D at Microsoft to speak with us. Crescent owner Ethan Winer conducted this interview, in which Greg discusses his involvement in the development of BASIC, and shares his insight on the current state of the art in compilers.

EW:
Greg, tell us about your background with Microsoft, and how you came to be involved with BASIC.

GW:
I started with BASIC in 1970, working with the systems software on an HP-2000 BASIC timesharing system. I eventually re-wrote about half the entire system, and at the same time it paid my way through undergraduate school. While I was in graduate school at Harvard working on a PhD in numerical analysis, I bought an Intelligent Systems CompuColor 8001 microcomputer. That's how I got started in microcomputing. I taught computer science for a year at Vanderbilt University, and then I joined Intelligent Systems as director of software. That was in 1978, and we ported a number of the Microsoft products to the operating system that was running on their machines. We converted the CP/M versions of BASIC, FORTRAN, and the macro assembler and linker. I went to work for Microsoft in 1979 and finished the 8080 version of their BASIC compiler. Then we got the contract for the IBM BASIC compiler.

Tim Patterson—author of the original MS DOS—and I worked on the BASCOM 1.0 compiler and runtime system. The first version of this compiler was fairly limited, really just a slight superset of BASICA.

EW:
What aspect of the BASCOM development did you find the most difficult?

GW:
Well, the runtime environment is very complex, because it is essentially a simulation of BASICA. BASICA is of course a total programming environment for BASIC. When you enter BASICA you could be in one of a variety of different screen modes, and it has to operate correctly from that state.

The compiler environment has to simulate this, unlike programs written in, say, FORTRAN or Pascal that have little interaction with the

actual hardware. Those languages do everything through the operating system. But when you start up a BASIC program, it has to find out what state the machine's in. There's a lot of hardware querying to find out where you're starting from.

EW:
What about QuickBASIC—was that a total rewrite of BASCOM?

GW:
No, it has really gone through a series of transformations. When we created IBM BASCOM 2.0 we added subprograms, some new graphics support, and we added large arrays. From that product we produced QuickBASIC 1.0, which is essentially a compiler without an environment. Then we did QuickBASIC 2.0, adding an editing environment written in C, which served as a shell to the compiler and runtime kept in memory. The compiler was not changed substantially for that release, aside from the ability to generate object code directly to memory.

QuickBASIC 3 is essentially another revision of that product. We added some more language features, and integrated debugging, as well as inline 8087 code generation.

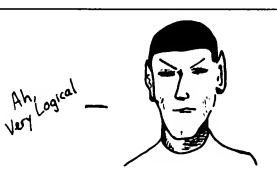
EW:
But QB 4 is obviously very different from the earlier versions.

GW:
Yes, with QuickBASIC 4 we added quite a few language features. We added data types, recursion, long integers, true functions, the interlanguage calling convention, and we significantly speeded up performance in other areas.

I think it's pretty easy to see that BASIC is a very popular applications programming language. We added these features to make BASIC an easier language to program in. We extended the control structures, for example adding Select/Case and Do statements, which are ANSI constructs. QuickBASIC 4 is a pretty major investment in manpower.

EW:
How many are people involved?

GW:
If you count me, there are nine of us. I'm not the project leader, though I occasionally do some of the compiler work when not managing the rest of Languages development. There's one person working on the compiler, three people developing the runtime package, and four more doing work on the interpreter.



Greg Whitten cont.

Lisp—the two other languages you'd consider in the same vein in terms of their execution speed. The ability to do an edit and then continue was a very difficult feature to implement, but once you've used it, you can't live without it!

EW:
What advances do you envision for QuickBASIC in the future?

GW:
We would like to make it easier to write applications programs, and to provide more expressiveness in the language. Of course supporting new operating environments is also important, and there is already a version of QuickBASIC 4 that runs under OS/2. Microsoft is definitely committed to making QuickBASIC the best BASIC available.

C

Crescent Software Products

A Quick Look

Below is a capsule description of each of our products, with a comprehensive discussion of their features following. If you have a particular programming problem and are not sure which product would be most appropriate, please write or call, and we will be happy to assist you.

Currently in progress is QuickEdit–a high-performance word processor that can be completely customized, and QuickMenu–a DOS menu that *doesn't* remain in memory.

QBase ($99)
QBase is a combination screen designer *and* full-featured relational database. Because it comes with complete BASIC source code, QBase can be customized to suit a developer's exact needs. Instructions are provided for modifying the various programs, and many pages in the documentation are devoted to an explanation of the system's internal workings.

We also provide a separate $Include file with routines to display screens and perform full-screen data entry and editing, for programmers who need only those capabilities. Example programs are included to illustrate the variety of ways QBase and its supporting routines may be incorporated.

Unlike other screen builders you may have seen, QBase does *not* require any special memory resident programs, and it doesn't access the disk each time a screen is to be displayed.

QBase Report ($69)
QBase Report enhances QBase, by offering multiple levels of sorting, grouping, record updating, browsing, and transaction posting. A simple command language is used to indicate the main file to report on, as well as any related files that are to be included. Capabilities include field calculations and totals, headers and footers, date arithmetic, printer codes, and chaining multiple reports.

TIP:

Until Microsoft gets around to adding a Merge capability to the QB editor, you can use this technique instead:

Load the file from which you want to copy into QuickBASIC's editor, and mark all or part of it as a block using the Shift-arrow keys. Next, press F2 to put the text into the clip board, and then load the program that the block is to be pasted into. Finally, put the cursor at the correct place in the file and press the Insert key. Voila–merged!

And while we've got Microsoft's ear–how about an On/Call statement? I envision it working like this:

```
On Choice Call EnterData,_
      Reports, Posting, Etc
Goto MainMenu
```

QuickPak ($69)
QuickPak is a collection of more than 65 "toolbox" routines for both beginning and advanced BASIC programmers. Approximately half are written in assembly language to perform functions that BASIC either cannot do directly, or is impossibly slow at. The remaining routines provide a variety of functions that would be difficult or tedious for many programmers to create themselves.

Included are programs for windowing, access to DOS and BIOS services, searching and sorting string arrays, creating pull-down and horizontal menus, accepting data input, and much more. PC Magazine gave QuickPak a glowing review (June 9, 1987), and we're convinced you will too.

QuickPak 2 ($49)
QuickPak Volume 2 contains more than thirty additional assembler and BASIC toolbox routines. Highlights include a multi-line text input subprogram with full word wrap, and a set of disk and printer tests that eliminate the need to use On Error in your programs. QuickPak 2 is currently available for use with QuickBASIC *only.*

Crescent Products Continued

GraphPak ($69)

GraphPak is a collection of subprograms and assembler routines for creating and displaying 3-D line, bar, and pie charts within your BASIC programs. It features manual or automatic scaling, as well as the ability to display titles and legends in any size or style. GraphPak also includes a font editor for creating customized character sets.

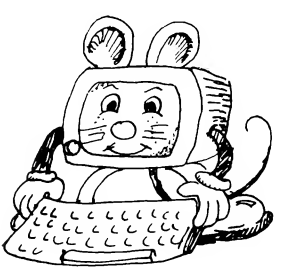
Program comments are often the most expressive part of a programmer's craft. We've seen programs with great ones and programs with none. Here's a few favorites:

'trust me - it'll work!

'Ouch!

'cook until done

'begin spel check



Walt Disney's P.C.

Celebrity P.C. # 87

Recommended:

We don't often recommend products from other companies, but this one is hot! Mode-MGA allows you to easily add Hercules graphics to your QuickBASIC programs. It also comes with a special memory-resident module that gives the same capability to *any* program.

To use Mode-MGA with QuickBASIC, simply add the supplied .OBJ file to your program when it is linked. To run games or other programs for which you don't have the source, simply execute the .COM version from the DOS prompt first—amazing!

Mode-MGA costs $79.95 (a royalty-free developer's package is available at extra cost), and can be ordered from

T.B.S.P.
8821 Alcott Street
Los Angeles, CA 90035

(213) 275-0208

Also highly recommended are pre-formatted disks. Why sit around for hours formatting disks yourself, when you can buy them ready to go for pennies more. We buy all of our disks from Ed Leach at

Data Dynamics
Suite 9
1854 South Coast Highway
Laguna Beach, CA 92651

(714) 494-8737

Order Form

QBase	QB( )	TB( )		$99.00 ea.
QBase Report	QB( )	TB( )		$69.00 ea.
QuickPak	QB( )	TB( )		$69.00 ea.
QuickPak Volume 2	QB( )			$49.00 ea
GraphPak	QB( )	TB( )		$69.00 ea.
QuickMenu	QB( )			$49.00 ea
Shipping and Handling (see below				
CT residents MUST add 7.5% sales tax				
Total Enclosed				

Please include shipping and handling charges as follows:

United States - $3 per order, $10 UPS Overnight
Canada - $3 per order 1st class, $15 UPS overnight
Europe - $10 per item airmail, $35 for 1 or 2 items UPS overnight
Anywhere else - $20 per item airmail

The fine print:

Separate versions are available for Microsoft QuickBASIC and Borland Turbo Basic--PLEASE specify when ordering. On request we can provide any product on 3-1/2 inch disks for use with the IBM PS/2 computers. Other 3-1/2 inch formats are also available.

Because we offer these products in two different versions (Turbo Basic and QuickBASIC), you may purchase the SECOND version for a substantial discount: QBase --$49, QBase Report, GraphPak, QuickPak--$39. New Manuals are not included in this offer.

We accept Visa, Master Card, C.O.D., checks, and of course cash. Purchase orders will not be accepted..

Please be sure to include your name and address with the order.

All Crescent Software programs come with full source code, and a 30-day satisfaction guarantee. We do NOT use copy protection. No royalties are ever required when using Crescent products in your own programs.

Overseas orders:
We now have three European distributors to assist you in purchasing our products. While we are always pleased to ship anywhere in the world, you may find it more convenient to contact a dealer in your area

In West Germany -

Ingenieur - Buro
Harald Zoschke
Berliner Str. 3
D-2306 Schoenberg/Holstein
West Germany - Telefon 04344/6166

In Holland -

LeMax Company B.V.
Robert Kockstraat 2
1171 BE Badhoevedorp
Holland - Telefoon 02968-4210

In Sweden -

Per Linderoth
Unitex
Malmvagen 28
19104 Sollentuna
Sweden

Comparing Call, Gosub, and multi-line functions

by Ethan Winer

BASIC offers a variety of ways to create subroutines, but which is the best approach for your programs?

With all the features and capabilities in the latest wave of BASIC compilers, we are often asked to explain the difference between a Called subprogram, a Gosubed routine, and a multi-line function. In many cases it really doesn't matter which you use, however some situations definitely favor one approach over the others. For example, using separately compiled subprograms is the only way to create a final program with more than 64K of code with QuickBASIC.

Essentially, a called subprogram would be used when you want to *do* something or need to alter a variable, such as the QuickPak TextIn or encryption routines. On the other hand, a function would be indicated when a value is to be computed based on the contents of one or more variables. Functions have the added characteristic that they cannot alter any of the incoming variables. Finally, Gosub routines are useful when a large number of variables must be shared with the main program. Let's take a closer look at each of these methods in turn.

SUBPROGRAMS

When a subprogram is called, the addresses of all the variables included in the parameter list are passed to the subprogram. Whenever one of these variables is assigned in the subprogram, those changes will be reflected in the main program because the variable was actually altered. Further, any of the variables used in the subprogram that were *not* passed in the list are considered to be "private". That is, you can have a variable named Price in the subprogram, and it will not conflict with a variable of the same name outside the subprogram.

Of course, you can also share variables between a main program and its subprograms with the Shared statement. There are two ways to do this. One approach is to declare the

variables as Shared within the subprogram. The only problem with this method is that in QuickBASIC a bug causes the Shared statement to be ignored once a program exceeds a certain size. We have observed this with QB versions through 3.0, so it isn't recommended.

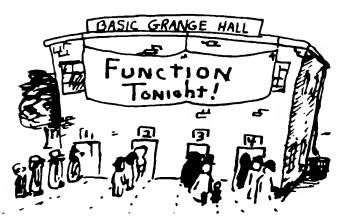
The other way to declare one or more variables as being shared throughout an entire program is with Dim Shared. Even though Dim is usually meant to be used with arrays, you can also use it to indicate which variables are to be global throughout an entire program. Here's one example:

```
Dim Shared Max, Position, Array$(250),_
     Zero, One
```

FUNCTIONS

Functions are useful in a variety of programming situations, and have the unique ability of returning a numeric result based on a string, or vice versa. For example, the QuickPak PUsing function accepts an incoming numeric value, and returns it as a formatted string:

```
X$ = FnPUsing$("###,.##", Value#)
```



Similarly, QuickPak's FnEval# accepts an incoming string, but returns its value after removing any punctuation:

```
Value# = FnEval#("$43,562.89")
```

Since a function can't actually change any of the variables being passed to it, it would be less appropriate for, say, modifying a string in some way. Of course, BASIC lets you do just about anything you'd ever want, however using a function that way would require extra program instructions. Had the QuickPak Ups routine been written as a function, it would have to be used like this:

```
CustName$ = FnUps$(CustName$)
```

As opposed to simply

```
Call Ups(CustName$)
```

If you wanted to use a capitalized version of a string, though, without actually changing it permanently, then a function would be ideal:

```
If FnUps$(Answer$) = "YES"  Then ...
```

or

```
Print FnUps$(CustName$)
```

This way, the string isn't actually altered, but your program can use it as if it were.

Functions are also ideal for providing a single result based on a number of input variables. Many of BASIC's

built-in functions work like this as well, for example Instr accepts two different string arguments, and returns one numeric result. Another place a function would be useful is when combining the foreground and background color bytes for use with the QuickPak video routines:

```
Def FnClr%(FG, BG) = (FG And 16) *_
    8 + (BG * 16) + (FG And 15)
```

Then you could call QPrint or APrint specifying the colors themselves, without needing the same formula each time:

```
Call QPrint(G$, FnClr%(15, 3))
```

Unlike subprograms where the incoming variables may be freely changed, BASIC instead makes a temporary copy of any parameters being passed to a function. Then, the addresses of these copies are made available to the function, rather than the addresses of the original variables. Thus, any variable assignments within a function will be in effect only while the function operates, since only the copies are being manipulated.

One final note is that unlike subprograms, variables that are not part of the function's incoming parameter list will be common to the rest of the program. Of course, functions can also utilize local variables, but they must first be declared Static.

GOSUB

In most cases, a Gosub is not as useful as a Called subprogram, since the names of each variable being manipulated must be whatever the subroutine is expecting. In the old days, the only way you could have a subroutine do the same thing to many different variables was to constantly assign and re-assign:

```
Temp$ = CustName$ : Gosub Ups
CustName$ = Temp$
```

One exception, though, is when a large number of variables must be shared between the main program and the subroutine. When I first began developing QuickEdit--a word processor I've been tinkering with for about a year--I started to write the margin setting routine as a subprogram. But then I was faced with either passing all the margins and other assorted default values as parameters, or else declaring the twenty or so variables as being Shared. It finally hit me that a plain old Gosub made the most sense. The routine could access all of the variables it needed, and I didn't have to pass or share any of them.

Gosub

Yet another advantage of Gosub is due precisely to the fact that it is not kept separate from the rest of the program. One of QuickBASIC's many restrictions in its error handling is that errors that occur in a subprogram must always be handled in the main program. If the routines that perform disk reading and writing are all in the main program to begin with, you'll have less difficulty resuming a program's execution where you want to.

TIP:

Here's a nifty way to clear the screen from the cursor to the end of the line using the QuickPak QPrint routine:

```
Call QPrint(Space$(81 - Pos(0)), 7)
```

And this will clear to the bottom of the screen:

```
Call QPrint(Space$(2001 - 80 * CsrLin - Pos(0)), 7)
```

Besides being incredibly fast, QPrint also leaves the cursor where it was. Of course, you can use any color you'd like.

News flash:

New Turbo Basic offers Herc graphics

Borland International has recently announced an improved version of their popular Turbo Basic compiler. Most important in this release is support for Hercules graphics, though according to Bob Zale—creator of Turbo Basic—a number of other improvements have been added as well. Turbo's file I/O time has been reduced, plus improvements have been made to the floating point library routines. Kudos to Bob and Borland!

Play "MSL16O1G>CEL12GP16L16EL3G"

An In-Depth Look at Crescent Software Products

You were expecting to get off easy?

QuickPak ($69)

QuickPak is a superb collection of enhancements, subroutines, and instructional material to help you get the most out of programming in BASIC. It includes expertly written assembly language routines to give your programs more speed, more power, and full access to all of the DOS and BIOS services.

In addition, QuickPak comes with a complete library of BASIC programs and functions, including input and menu routines, date and time arithmetic, and a wealth of practical and useful examples. But QuickPak is more than just programs, because you'll also learn how to create your own assembler routines and extensions.

Written from a BASIC programmer's perspective, The Assembly Tutor shows how easy learning assembly language can really be. It begins by comparing BASIC commands with their assembler counterparts. Then, using clear and practical examples it progresses to passing variables, interfacing with BASIC, creating source programs, and more. Clever illustrations emphasize the topics discussed, to help you *really see* what assembly language is all about.

And then there's the QuickPak Tips & Tricks book—packed with programming tips, useful techniques, and interesting quirks about BASIC. Beginner or advanced, you'll be enlightened and amused.

All of the programs in the QuickPak collection are designed for easy use—absolutely *no* advanced programming knowledge is required. With QuickPak you can forget compli-

QuickPak comes with all of the source code for every program included on the disk.

cated BLoads, Pokes, and endless Data statements. For example, to perform a "print screen" from within a running BASIC program, merely add this single line:

```
Call PrtSc
```

It's as easy as that! Further, QuickPak subroutines are added to your programs when they are compiled and linked, so no additional utilities are needed at run time. But there's more. QuickPak comes with all of the source code for *every program* included on the disk. Use these programs as is, or change them if you prefer. Each source listing contains extensive comments that explain how it works and why.

DOS Services:

- **BasDir** and **ReadDirs** will read directory and file names from any drive or path into a string array. You can use the DOS wild cards to include only certain files.

- **FCount** and **DirCount** complement BasDir and ReadDirs by providing a count of all files or directories that match a given search specification.

- **DosVer** reports which version of DOS is currently installed. This is essential when writing applications that use QuickBASIC's network capabilities.

- **Exist** mimics the DOS EXIST command, and provides an easy way to tell if a given file is present before attempting to read it. Other methods require either convoluted error handling, or create empty files in the process.

- **FileInfo** provides a complete report for any file, giving its size, date, time, and attributes.

- **GetDir** returns the current directory. BASIC provides an easy way to *change* directories, but no means to tell which is the current default.

- **GetDrive** and **SetDrive** allow you to read or set the currently logged disk drive.

- **GetSpace** will quickly analyze any floppy or hard disk, and return the sector and cluster makeup, along with the total and free space remaining.

- **GetVol** retrieves the volume label from a disk.

- **ReadSect** and **WritSect** will read or write any sector on any disk. These are extremely powerful utilities, and could become the foundation for a complete disk editing program.

- **SysTime** returns the current system time to the hundredth of a second. If you ever need to know the *exact* time, this is the best way to get it.

- **WeekDay** calls upon DOS to return the day of the week (1-7) for a given date.

String Services:

- **Find** will search an entire array for any string or sub-string *regardless of capitalization* Find also accepts wild cards to match any character.

- **Sort** will alphabetize all or just a portion of a string array at incredible speed.

- **Ups** provides a convenient way to quickly capitalize a string.

Video Services:

■ **QPrint** can display any string in a fraction of the time it takes BASIC. **APrint** is also for quick printing, but it displays any portion of an *array* and contains it within a specified area of the screen. APrint greatly simplifies creation of a "browse" facility, where an entire string array may be scrolled up or down, left or right.

■ **LScroll** and **RScroll** directly scroll portions of the screen either left or right. **Wind** and **Wind2** complement these scroll routines by allowing you to clear a window on the screen to any color, or scroll it up or down.

■ **PaintBox** can color any portion of the screen, but *without* requiring the text to be re-printed. Simply specify the top and bottom corners, and PaintBox does the rest.

■ **PrtSc** will perform a "print screen" from within a program, just as if the PrtSc key had been pressed.

■ **ScrnSave** and **ScrnRest** are used to save or restore any area of the screen, and allow you to incorporate windowing within a BASIC program. Simply call ScrnSave *before* you print any help or other messages, and ScrnRest will bring everything back the way it was when you're done.

Input Routines:

All of the QuickPak input routines are intended as replacements for the BASIC Input command. Where Input doesn't allow you to edit an existing string, or control the length and colors, these subprograms do it all.

Another advantage is that an exit code is returned, to let you know how editing was terminated. For example, if someone is entering data and presses the up arrow key, they would expect the program to move up to the previous field. All of these routines provide a consistent way to determine which keys were pressed.

Finally, each input routine will display the current status of the CAP and NUM lock keys at the bottom of the screen. Of course you may change where the status is displayed, or disable it entirely if you prefer.

■ **TextIn** is the main input routine in this collection. Besides allowing you specify how long a string may be entered, TextIn lets you optionally specify that letters are to be converted to upper case, or that only numbers are acceptable.

■ **DateIn** and **NumIn** are additional input routines, but are specialized for date and numeric input respectively.

DateIn automatically provides and skips over the separating slashes for date fields, while NumIn allows numeric values to be easily entered or edited.

- **YesNo** provides a simple way to accept a "Y" or "N" choice, and automatically converts the response to upper case.

Menu Programs:

Four separate menu routines are included with QuickPak. Two provide the familiar "scroll bar" method for selecting items. The others use the Lotus 123 approach for displaying choices and prompts on a single line. Most adjust their size *automatically*, based on the number and length of the choices.

A complete pull-down multiple menu subprogram is also included. It operates just like the menus used by the QuickBASIC and Turbo Basic editors. Simply define your list of choices for each menu, and Pull-Down does the rest.

Date and Time Arithmetic:

Two subprograms are provided for converting dates to numbers and back, and two others report the day of the week from either format. Once the dates have been converted to the equivalent numbers, adding or subtracting them is trivial.

FnAddT$** and **FnSubT$ complement the date routines above, by allowing you to add or subtract any two times (24 hour basis).

Miscellaneous Calls and Functions

Encrypt and **Decrypt** comprise a simple but effective way to protect your sensitive data.

QSort1 and **QSort2** augment the original QuickPak Sort routine, except QSort1 is meant for numeric arrays, and QSort2 sorts strings based on a starting and ending portion of each element.

FnBigMod works just like BASIC's Mod, except it works with any numbers. (Mod crashes with numbers beyond 32767.)

FnEval# takes the value of any string expression, ignoring dollar signs and punctuation.

FnLastFirst and **FnLastLast** provide a simple way to swap first and last names. For example, they will convert "Smith, John" to "John Smith" and vice versa.

FnLCount will quickly count the number of lines in a text file. The only other way would be to read in each line one by one, which takes forever.

FnPad$ pads a number with leading zeros to any specified length.

FnPack$** and **FnUnPack$ are used to pack any date to only three bytes, saving disk and memory space. **FnParse$** and **FnUnParse$** will quickly convert any string of digits to the equivalent ASCII printer codes.

FnPUsing$ works just like BASIC's Print Using, except it returns the formatted value as a string.

FnTrim$ strips a string of leading and trailing blanks.

QuickPak is **EASY TO USE**

To include QuickPak, merely call up QuickBASIC like this:

```
QB /L QuickPak.Exe
```

Most of the QuickPak routines are accessed with the BASIC Call command. Below are some actual examples showing how simple QuickPak is to use.

Read a directory into a string array:

```
Call BasDir(Array$(0))
```

Sort an entire string array:

```
Call Sort(Array$(0), Size%)
```

Enter/Edit a string of controlled length:

```
Max = 20 : Call TextIn(T$)
```

Display a string instantly on the screen:

```
Call QPrint(X$, Colr%)
```

QuickPak is **POWERFUL**

- Sort a 30K text file in less than three seconds on a stock IBM PC—even faster on an AT or PS/2.

- Find the 2000th name in a list in *less than a second.*

- Quick Print a full screen of text on a monochrome or EGA screen so fast that it *can't be timed* — a CGA display takes less than 1/4 second. Compare this with 2-1/2 seconds for QuickBASIC's regular Print command.

QuickPak Volume 2 ($49)

QuickPak 2 picks up where QuickPak leaves off, offering many new and enhanced assembler and BASIC functions. It is currently available for use with QuickBASIC *only*, and includes:

- A multiple line text input routine with full word-wrap capabilities. **TextIn3** allows you to place a mini word processor of any size anywhere on the screen within your programs.

- New versions of **QPrint** and **APrint** that can write to any of the CGA or EGA text screen pages.

- An enhanced window routine that lets you save and restore any number of screens.

- True Binary file access! A collection of assembler subroutines that allow you to read and write *any* portion of any file directly, without stumbling over the DOS end-of-file marker, or having to deal with a zillion one-byte records. Also included is an example program that lets you copy files from BASIC without requiring Shell or COMMAND.COM.

- A trio of routines to eliminate the need for On Error in your programs. Many people like to avoid On Error, because it makes a BASIC program larger and run more slowly. **ReadTest** reports if a drive is ready for reading, and **WritTest** checks it for writing. Between the two you can quickly determine whether the drive door is open, or a disk is write-protected. **PRNReady** is a BASIC example showing how to tell if a printer is on-line and available.

- A modified version of the QuickPak Find that allows searching forward as well as backward.

- A pair of routines to easily get or set a file's attribute.

- An assembler version of Quick Sort for string arrays, only this one sorts a parallel integer array which serves as an index into the strings. It's great for maintaining a table of disk record numbers, or any other place where access to a list in sorted order is needed, without physically changing it.

- A vertical menu subprogram that handles more choices than will fit on the screen at one time. It recognizes PgUp, PgDn, Home, End, and even the first letter of the choice.

- A compliment to QuickPak's GetVol–**PutVol** lets you add or change the volume label on any disk or hard drive.

- Some wonderful contributions from QuickPak users include two assembler routines that stay resident in memory, but only while your program is running. **Clock** displays the current time in the location and color of your choice, while **Keyboard** provides a CAP or NUM message when those keys are activated. The beauty of these routines is that your BASIC program controls when they are installed and removed, but *never* has to poll the keyboard or the system clock to create a continual display.

- Other submissions include a routine to draw any type of box very quickly, a complete horizontal menu for selecting file names much like the one QuickBASIC uses, and a compact version of TextIn. All of these are written in assembly language, and come complete with fully commented source code.

QBase ($99)

QBase is a revolutionary concept in BASIC screen and database software, not only because it's so easy to use, but also because it does so much. Where most screen builders merely create a screen image to be displayed by your program, QBase lets you define data entry fields, create custom help screens, and even manage an entire *relational* database.

This means you can keep a list of customers in one database, and a series of transactions in another. Then, whenever a sale is entered, the program will retrieve the name and address from the customer file automatically. Relations may be established between any files in the database, based on a match between like fields.

QBase can do all this because it's really two separate programs–the screen builder and a run-time application. If you prefer to write your own routines to handle file I/O and indexing, then skip the run-time program and simply use QBase for screens and data entry. It's still a great deal. But the database is so powerful and complete, it would be a shame not to use it.

What it isn't

QBase is *not* a code generator. The main problem with code generators is that they can never do exactly what you need. So after they create a program, you'll undoubtedly have to make changes. But once you've modified the code, it's very difficult to go back and alter the original screen, because you'll have to add your changes all over again. Further, most BASIC code generators create programs that are poorly organized and use line numbers.

The QBase approach

QBase avoids that mess by saving screens and fields as *form definitions.* The definitions are logically arranged and kept in disk files, which are simple to access from any program. Multiple screen definitions are contained in a single library file, which is loaded once at the start of a data entry session. Then as each individual screen is needed, it can be displayed very quickly. Other libraries may be loaded when needed, allowing an unlimited number of screens in a single program.

But the real advantage of QBase is its run-time application. Since a single program is used for all of the databases, any change you make to the run-time module will be reflected in the applications you create. If you wanted to re-assign the function keys or the location of the menus, you only need to change one program.

This means you can modify an *existing* database, without having to

go back to the original screens. And, if you ever need to add, change, or delete fields, a "re-build" facility is included that will convert an existing data file to the new format automatically.

How it works

The screen builder features pull-down menus, multiple help screens, and function keys that are logically organized. Separate menus are used to load and save screens, select colors and line types, draw boxes, and cut or copy blocks. Shortcut keys are also provided for most of the operations if you prefer.

Defining fields is as simple as pushing a key, and answering a few questions. You can specify string, integer, single, or double precision field variables, as well as date, phone, social security, zip, currency, logical, and multiple choices.

The multiple choice fields are especially useful, because they eliminate extra typing and spelling errors. For example, if you're creating an accounting system that must keep track of expenses by category, most databases would require you to define the category field as text.

Now suppose the data entry clerk types "Payroll" one day, but goofs the next time and enters "payrll". Good luck trying to balance your books!

QBase instead lets you establish a list of acceptable choices, which are displayed in a pop-up menu when the cursor is on that field. Only the choices shown will be accepted, and even better, the choice number will be stored in the disk file as a single Chr$() byte. This provides a tremendous reduction in disk space compared with storing the full text of each choice.

The QBase run-time program is truly a full featured database. It allows stepping forward and backward through the file, searching based on multiple criteria, and adding and updating records. More than 15 screens can be included in a single database, and each screen may contain up to 100 fields and 32,000 records. A standard help screen is always available, though it may be edited just like any other QBase screen. You can also create custom help windows for each entry screen.

What's included

QBase comes with a screen builder, run-time and re-build programs, a complete help window system, sample screens and applications, and even has a slide show program which displays a series of screens automatically.

QBase also includes fully commented BASIC source code to show how it works, and where it may be customized. The assembler source

requires QuickPak, which is available separately.

And as a special bonus, we've added The Hardware Tutor—a clear and practical discussion of electronic concepts and circuits, written from a programmer's perspective.

The bottom line

QBase costs only $99, which includes a license to distribute unlimited copies of the database and re-build programs, as long as they are only used to augment applications you develop with QBase. QBase requires DOS 2 or later, and a minimum of 256K memory. If you plan to modify and recompile the BASIC source, then at least 512K is needed.

QBase field types

String
Upper Case String
Numeric String
Integer
Single Precision
Double Precision
Currency
Date
Phone Number
Social Security Number
Zip Code
Multiple Choice
Logical

QBase maximum limits

Records per file	32,766
Fields per form	100
Indexes per form	5
Files open at once	5
Relational fields	no limit

QBase performance tests

The following tests were performed on a stock IBM AT with a 30MB hard disk, using a "real-life" data file comprised of 21,000 records. All timings indicate how long it took to locate the last record in the file for each field type using a typical search value. For smaller files, the times will be proportionally less.

The actual search times will vary for string fields, depending on the number of similar records. For example, locating the last "Jones" will generally take longer than finding the last "Szabo", because more records have to be examined. Notice that when date or integer fields are used (part numbers, account numbers, etc.), the response time is nearly instantaneous.

Indexed Searches:	**Seconds**
Using an integer field	0.03
Using a date field	0.03
Using a string field	10.00
Non-Indexed Searches:	
Using any field type	77.00

QBase Report ($69)

QBase Report offers three kinds of reporting capabilities. The conventional report type lets you summarize the contents of a file sorted on two key fields, with its output directed to the screen, a printer, or disk file. The Modify report type allows updating selected records in a file, for example flagging accounts that are more than 30 days overdue. The Add reporting option can be used to copy all or selected records from one data file into another. If needed, field type conversion will be performed automatically during record adding or modification.

A simple command language is used to indicate the primary file for the report, as well as any related files to be included. An "Assist" mode is also available to guide you through the report's design. Among the many reporting options are headers, footers, margins, current date, time, and page number, and even running totals. Totals from related files can also be included in a report, along with constants, and four-function math.

QBase Report lets you chain multiple reports to perform several functions in sequence without any user intervention. A unique Browse mode features a clever multitasking scheme, whereby the selected and sorted data may be browsed even while it's still being processed!

An on-screen report layout feature lets you quickly indicate how the report is to appear, and where each indicated field will be positioned. You may also specify that the report is to pause before running, allowing the user to specify either the screen, printer, or disk file.

QBase Report includes three additional bonuses—a complete time billing application designed for computer professionals, a stand-alone sort utility for use with any random files, and an import facility for QBase data files.

GraphPak ($69)

GraphPak is a comprehensive collection of subprograms for creating regular *and* 3-D bar, pie, and line graphs. It is comprised of both high-level routines and a number of low-level primitives. The high-level routines are responsible for accepting the numeric data and text to be displayed, and then perform the appropriate scaling, angle calculations, and color selection automatically.

Individual low-level routines are accessible to create textured backgrounds, position and print titles and legends, and perform scrolling and windowing in the popular CGA and EGA graphics modes. By supplying both high and low-level functions, the BASIC programmer is presented with the ability to display complex

graphs very quickly with a simple Call, as well as complete control when necessary.

Important features include the ability to display titles and legends at any location on the screen and select from one or more background tile patterns. Text may be scaled vertically or horizontally, tilted, and rotated. GraphPak also includes a full-featured Font editor for creating and editing custom fonts. A unique Tiling editor permits you to easily design your own background and fill textures in any color, plus a callable subprogram to send hi-res EGA screens to a LaserJet printer

High level routines provided:

- **PlotArray2** and **PlotArray3** accept the arrays containing numeric data and text legends, and create complete 2 or 3 dimensional graphs automatically.

- **PieChart2** and **PieChart3** are similar to the PlotArray routines, but instead draw and title pie charts.

- **PlotLine2** and **PlotLine3** draw 2 and 3 dimensional line graphs.

Low level routines include:

- **DrawAxis2** and **DrawAxis3** draw a 2 or 3-D background axis.

- **DrawBack2** and **DrawBack3** call upon the DrawAxis routines to provide a background axis, and then fill it with the tile patterns you specify.

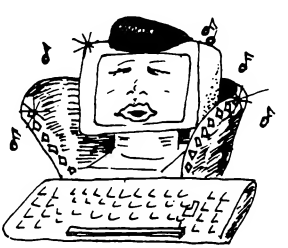
- **DrawChar** draws a single character at pixel location *x,y* in any size, shape, color, and at any angle.

- **DrawString** calls upon DrawChar to display a complete string with all of the above options.

- **Scroll1 - Scroll4** scroll the graphics screen either up or down, left or right.

- **GSave** and **GRest** save and restore selected portions of the graphics display memory for windowing and other purposes.

- **GetVideo** and **SetVideo** determine the type of display adpater present, EGA information (if installed), and the highest resolution possible for a given monitor at run time.



Elvis' P.C.

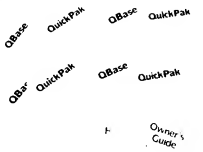
```
Play "MNL12O2GGEAL6GE"
```

Celebrity P.C. # 159

BASIC Programming Tools

There's nothing basic about these professional programming utilities.

Whether you're a seasoned expert or just starting out, we can help you create programs that run faster, work harder, and simply look better. We have built our reputation on customer satisfaction by providing expert advice and quality technical support. All Crescent Software products include source code, demonstration programs, clear documentation, and a 30-day satisfaction guarantee.



- **QBase** is a superb screen designer *and* full-featured relational database. Because we include complete BASIC source code, QBase can be customized to suit your needs. Besides its database capabilities, QBase is outstanding for creating custom titles, help screens, and product demos, and includes a versatile slide-show program. **$99**

NEW!

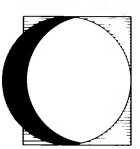
- **QBase Report** enhances QBase by generating reports with multiple levels of sorting, automatic record updating, browsing, and transaction posting. As a special bonus, QBase Report includes a complete, ready to run, time billing application for computer professionals. **$69**

- **QuickPak** contains more than 65 essential routines for BASIC programmers. Included are programs for windowing, access to DOS and BIOS services, searching and sorting string arrays, creating pull-down and Lotus™ menus, accepting data input, and much more. QuickPak is loaded with examples and tutorial information, and comes with a clever tips and tricks book, plus *The Assembly Tutor* — a complete guide to learning assembly language from a BASIC perspective. **$69**

- **By Customer Demand — QuickPak II**
  More than 30 additional tools, including disk and printer tests to eliminate the need for On Error in your programs, and a multi-line text input routine that lets you put a note pad with full word wrap anywhere on the screen from within your programs. Other routines include binary file access, more menus, multiple screen save and restore, continuous time display, automatic box drawing, and much more. (Available for use with QuickBASIC *only*.) **$49**

NEW!

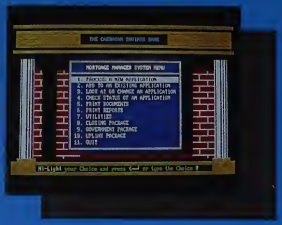
- **GraphPak** is an extensive collection of routines for displaying line, bar, and pie charts automatically within your programs. It will create 3-D charts with manual or automatic scaling, titles and legends in any size or style, as well as scrolling and windowing *in graphics*. GraphPak also comes with a sophisticated font editor for customizing your own character sets. **$69**



CRESCENT SOFTWARE

64 Fort Point Street, East Norwalk, CT 06855
(203) 846-2500

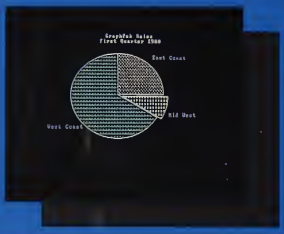
Separate versions are available for Microsoft QuickBASIC and Borland Turbo Basic — *please* specify when ordering. No royalties, not copy protected, of course. We accept Visa, M/C, C.O.D. and Checks. Add $3 shipping and handling, $10 overnight and foreign, $25 2nd day foreign.



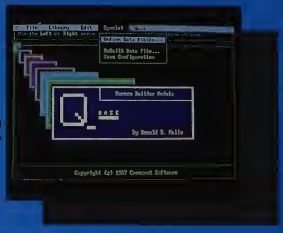
Screens!!!

Screens!!!

Screens!!!



Screens !!!



Crescent Software
11 Grandview Avenue
Stamford, CT 06905

(203) 846-2500

BULK RATE
U.S. POSTAGE
PAID
NORWALK, CT 06856
PERMIT NO. 282